ASSIGNMENT - 3

SDN and Ryu

Advanced Computer Networks

COL724

Submitted to:

Dr. Tarun Mangla
Assistant Professor,
Department of Computer Science and Engineering,
Indian Institute of Technology, Delhi

Submitted by:

Arun Malik,
Entry No: 2023EEZ8340
Ph.D. Researcher
Department of Electrical Engineering,
Indian Institute of Technology, Delhi

Objective: The objective of assignment was to obtain hands-on experience with Ryu, and learn how to implement specific network policies using OpenFlow-like APIs.

Environment Used: In this assignment, a fresh installation of Ubuntu 20.04 LTS was used with mininet and Ryu, along with their dependencies. Other tools used for the assignment are:

1. **iperf:** iperf is a tool for active measurements of the maximum achievable bandwidth on IP networks. It supports tuning of various parameters related to timing, protocols, and buffers. For each test it reports the measured throughput / bitrate, loss, and other parameters. Tests for both TCP and UDP based transfers have been performed as a part of throughput tests in the assignment.

Q1. The following topology of the virtual network is created by running the script ***assignment3_nw_topology.py*** with the following command in the bash terminal:

```
$ sudo mn --custom /home/arun/mininet/assignment3_nw_topology.py --topo
assignment3_nw_topology –controller remote –switch ovsk
```

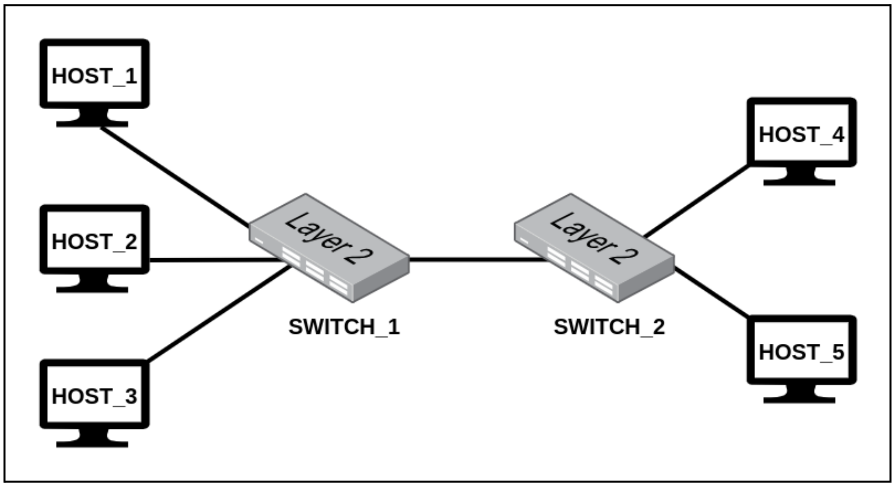
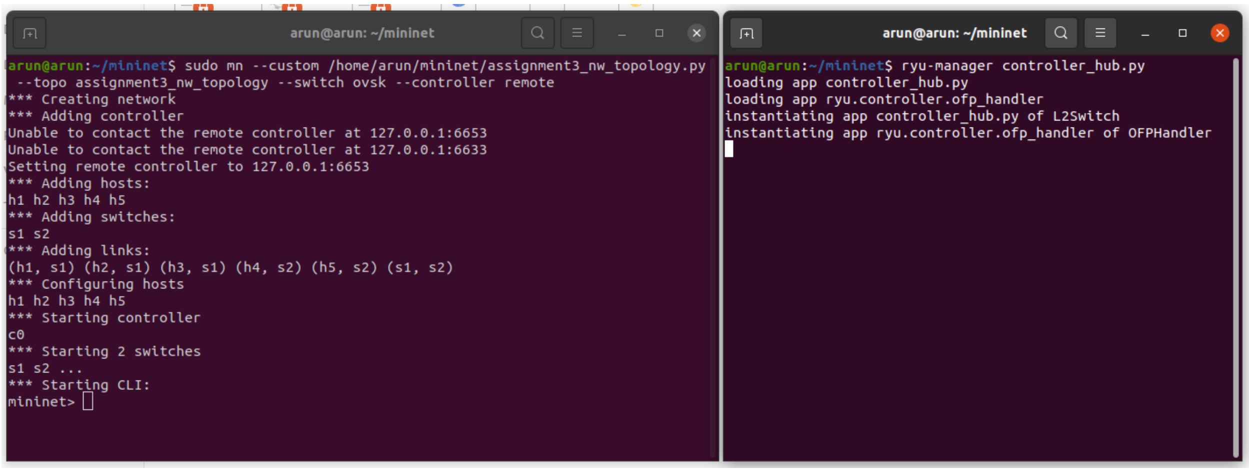


Fig.1, Virtual network topology

The parameter ***–custom*** specifies that a custom topology is being used. The parameter ***–topo*** is used to refer to the topology object name in the script.

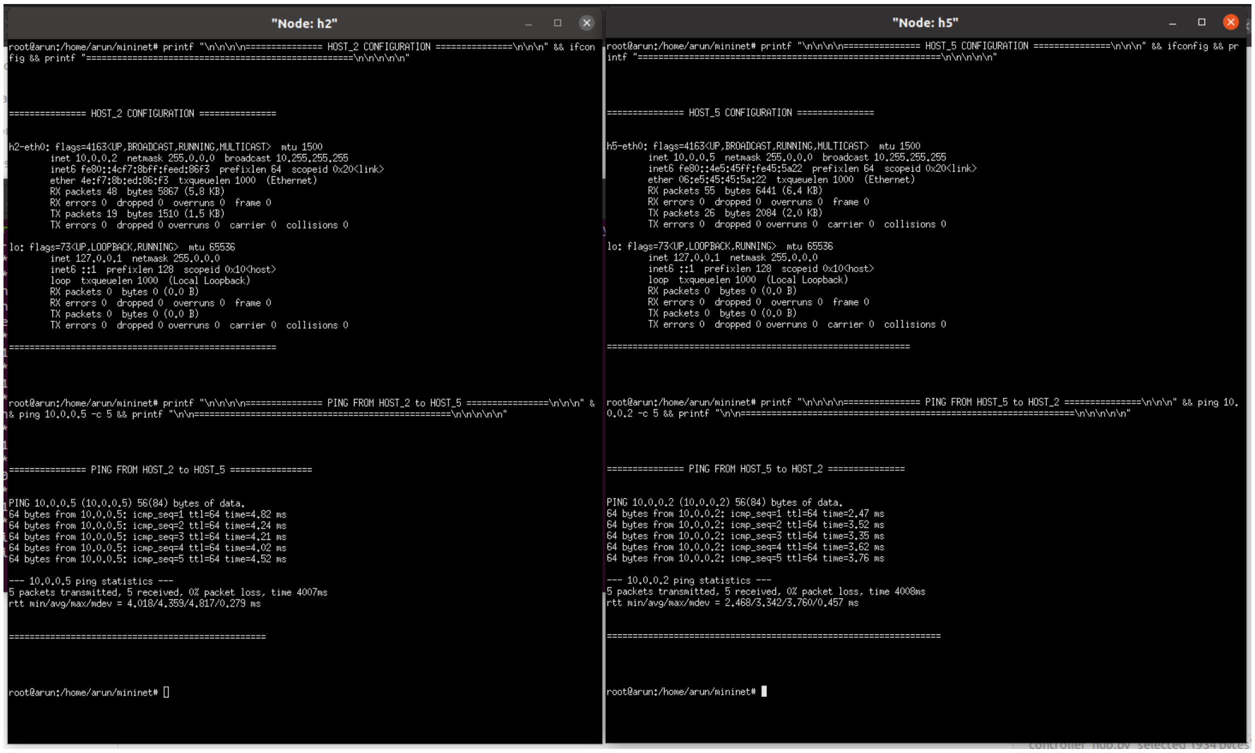
For the normal switch hub, there was no improvement in the ping response times (between h2 and h5) (Fig. 2 & 3).



```
arun@arun:~/mininet$ sudo mn --custom /home/arun/mininet/assignment3_nw_topology.py
 --topo assignment3_nw_topology --switch ovsk --controller remote
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6633
Setting remote controller to 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s2) (h5, s2) (s1, s2)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
mininet> 
```

```
arun@arun:~/mininet$ ryu-manager controller_hub.py
loading app controller_hub.py
loading app ryu.controller.ofp_handler
instantiating app controller_hub.py of L2Switch
instantiating app ryu.controller.ofp_handler of OFPHandler
```

Fig. 2

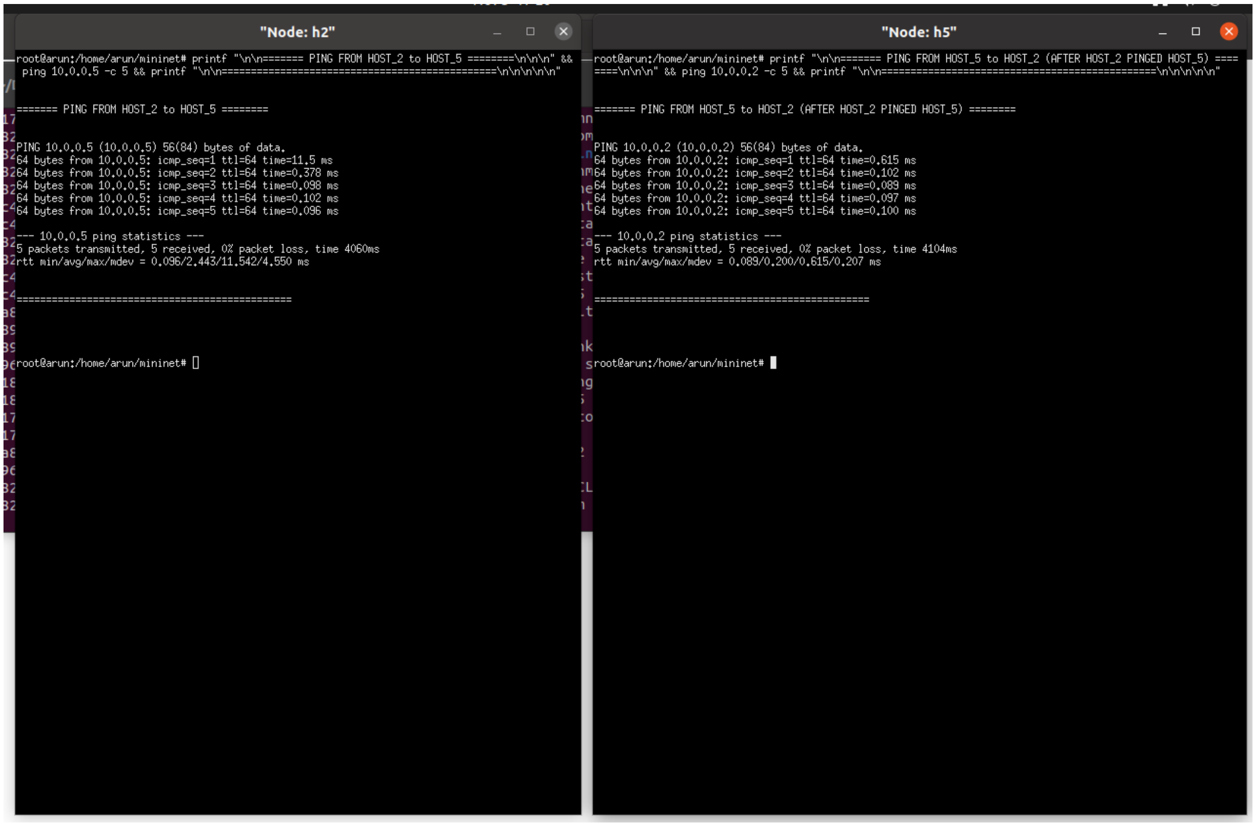


```
root@arun:/home/arun/mininet# printf "\n\n\n\n============== HOST_2 CONFIGURATION ==============\n\n\n" && ifcon
fig && printf "==================================================\n\n\n\n\n"




============== HOST_2 CONFIGURATION ==============


h2-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.2  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::4cf7:8bff:feed:86f3  prefixlen 64  scopeid 0x20<link>
        ether 4e:f7:8b:ed:86:f3  txqueuelen 1000  (Ethernet)
        RX packets 48  bytes 5867 (5.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 19  bytes 1510 (1.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

==================================================




root@arun:/home/arun/mininet# printf "\n\n\n\n============== PING FROM HOST_2 to HOST_5 ===============\n\n\n" &
& ping 10.0.0.5 -c 5 && printf "\n\n==================================================\n\n\n\n\n"




============== PING FROM HOST_2 to HOST_5 ===============


PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=4.82 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=4.24 ms
64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=4.21 ms
64 bytes from 10.0.0.5: icmp_seq=4 ttl=64 time=4.02 ms
64 bytes from 10.0.0.5: icmp_seq=5 ttl=64 time=4.52 ms

--- 10.0.0.5 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 4.018/4.359/4.817/0.279 ms


==================================================




root@arun:/home/arun/mininet# 
```

```
root@arun:/home/arun/mininet# printf "\n\n\n\n============== HOST_5 CONFIGURATION ==============\n\n\n" && ifconfig && pr
intf "============================================================\n\n\n\n\n"




============== HOST_5 CONFIGURATION ==============


h5-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.5  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::4e5:45ff:fe45:5a22  prefixlen 64  scopeid 0x20<link>
        ether 06:e5:45:45:5a:22  txqueuelen 1000  (Ethernet)
        RX packets 55  bytes 6441 (6.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 26  bytes 2084 (2.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

============================================================




root@arun:/home/arun/mininet# printf "\n\n\n\n============== PING FROM HOST_5 to HOST_2 ==============\n\n\n" && ping 10.
0.0.2 -c 5 && printf "\n\n============================================================\n\n\n\n\n"




============== PING FROM HOST_5 to HOST_2 ==============


PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=2.47 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=3.52 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=3.35 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=3.62 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=3.76 ms

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4008ms
rtt min/avg/max/mdev = 2.468/3.342/3.760/0.457 ms


============================================================




root@arun:/home/arun/mininet# 
```

Fig. 3

For the learning switch, the response times significantly improved after the first ping response, since the switch learned the path between h2 and h5 (Fig. 4).

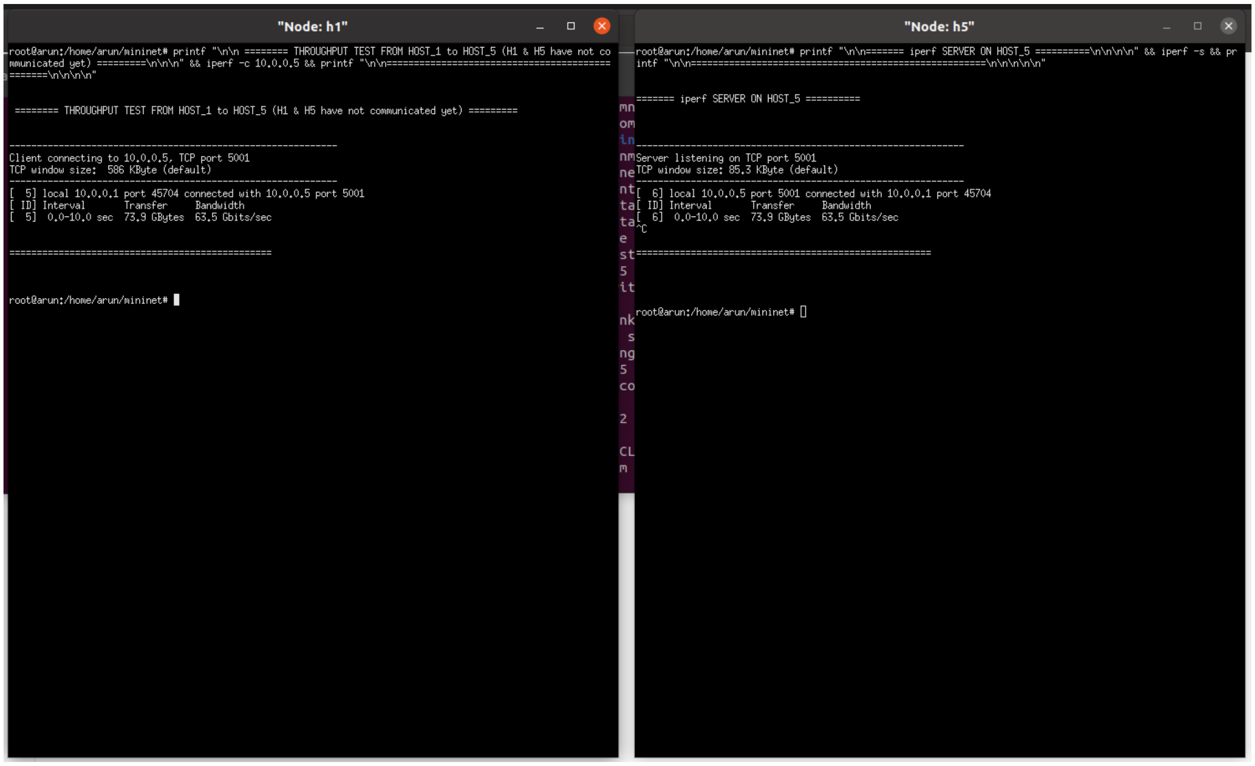


```
root@arun:/home/arun/mininet# printf "\n\n======= PING FROM HOST_2 to HOST_5 ========\n\n\n" &&
 ping 10.0.0.5 -c 5 && printf "\n\n=============================================\n\n\n\n\n"


======= PING FROM HOST_2 to HOST_5 ========


PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=11.5 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=0.378 ms
64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=0.098 ms
64 bytes from 10.0.0.5: icmp_seq=4 ttl=64 time=0.102 ms
64 bytes from 10.0.0.5: icmp_seq=5 ttl=64 time=0.096 ms

--- 10.0.0.5 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4060ms
rtt min/avg/max/mdev = 0.096/2.443/11.542/4.550 ms


=============================================



root@arun:/home/arun/mininet# 
```

```
root@arun:/home/arun/mininet# printf "\n\n======= PING FROM HOST_5 to HOST_2 (AFTER HOST_2 PINGED HOST_5) ====
====\n\n\n" && ping 10.0.0.2 -c 5 && printf "\n\n=============================================\n\n\n\n\n"


======= PING FROM HOST_5 to HOST_2 (AFTER HOST_2 PINGED HOST_5) ========


PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.615 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.102 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.089 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.097 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.100 ms

--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4104ms
rtt min/avg/max/mdev = 0.089/0.200/0.615/0.207 ms


=============================================



root@arun:/home/arun/mininet# 
```

Fig. 4

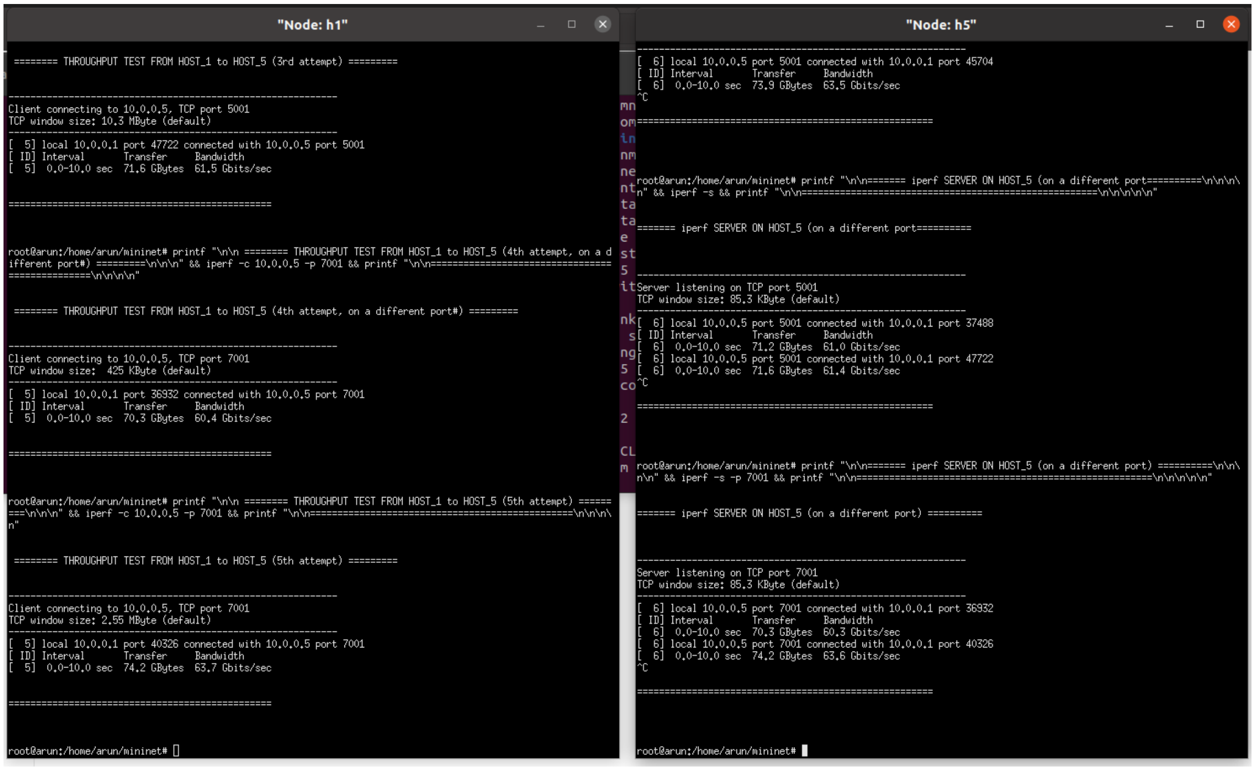
The throughput test between h1 and h5 showed no significant changes with respect to learning switch (Fig. 5 & 6)



```
root@arun:/home/arun/mininet# printf "\n\n ======== THROUGHPUT TEST FROM HOST_1 to HOST_5 (H1 & H5 have not co
mmunicated yet) =========\n\n\n" && iperf -c 10.0.0.5 && printf "\n\n========================================
=======\n\n\n\n"


 ======== THROUGHPUT TEST FROM HOST_1 to HOST_5 (H1 & H5 have not communicated yet) =========


------------------------------------------------------------
Client connecting to 10.0.0.5, TCP port 5001
TCP window size:  586 KByte (default)
------------------------------------------------------------
[  5] local 10.0.0.1 port 45704 connected with 10.0.0.5 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-10.0 sec  73.9 GBytes  63.5 Gbits/sec


================================================



root@arun:/home/arun/mininet# 
```

```
root@arun:/home/arun/mininet# printf "\n\n======= iperf SERVER ON HOST_5 ==========\n\n\n\n" && iperf -s && pr
intf "\n\n======================================================\n\n\n\n\n"


======= iperf SERVER ON HOST_5 ==========




------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  6] local 10.0.0.5 port 5001 connected with 10.0.0.1 port 45704
[ ID] Interval       Transfer     Bandwidth
[  6]  0.0-10.0 sec  73.9 GBytes  63.5 Gbits/sec
^C

======================================================



root@arun:/home/arun/mininet# 
```

Fig. 5

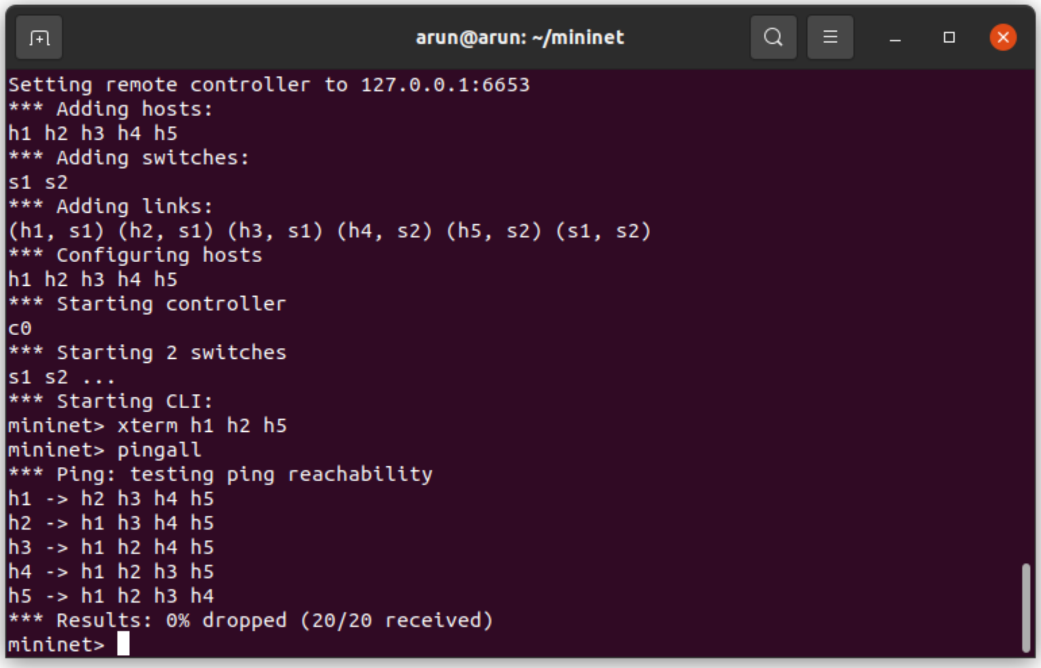


```
 ======== THROUGHPUT TEST FROM HOST_1 to HOST_5 (3rd attempt) =========


------------------------------------------------------------
Client connecting to 10.0.0.5, TCP port 5001
TCP window size: 10.3 MByte (default)
------------------------------------------------------------
[  5] local 10.0.0.1 port 47722 connected with 10.0.0.5 port 5001
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-10.0 sec  71.6 GBytes  61.5 Gbits/sec


================================================



root@arun:/home/arun/mininet# printf "\n\n ======== THROUGHPUT TEST FROM HOST_1 to HOST_5 (4th attempt, on a d
ifferent port#) =========\n\n\n" && iperf -c 10.0.0.5 -p 7001 && printf "\n\n=================================
==============\n\n\n\n"


 ======== THROUGHPUT TEST FROM HOST_1 to HOST_5 (4th attempt, on a different port#) =========


------------------------------------------------------------
Client connecting to 10.0.0.5, TCP port 7001
TCP window size:  425 KByte (default)
------------------------------------------------------------
[  5] local 10.0.0.1 port 36932 connected with 10.0.0.5 port 7001
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-10.0 sec  70.3 GBytes  60.4 Gbits/sec


================================================



root@arun:/home/arun/mininet# printf "\n\n ======== THROUGHPUT TEST FROM HOST_1 to HOST_5 (5th attempt) ======
===\n\n\n" && iperf -c 10.0.0.5 -p 7001 && printf "\n\n================================================\n\n\n\
n"


 ======== THROUGHPUT TEST FROM HOST_1 to HOST_5 (5th attempt) =========


------------------------------------------------------------
Client connecting to 10.0.0.5, TCP port 7001
TCP window size: 2.55 MByte (default)
------------------------------------------------------------
[  5] local 10.0.0.1 port 40326 connected with 10.0.0.5 port 7001
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-10.0 sec  74.2 GBytes  63.7 Gbits/sec


================================================



root@arun:/home/arun/mininet# 
```

```
------------------------------------------------------------
[  6] local 10.0.0.5 port 5001 connected with 10.0.0.1 port 45704
[ ID] Interval       Transfer     Bandwidth
[  6]  0.0-10.0 sec  73.9 GBytes  63.5 Gbits/sec
^C

======================================================




root@arun:/home/arun/mininet# printf "\n\n======= iperf SERVER ON HOST_5 (on a different port==========\n\n\n\
n" && iperf -s && printf "\n\n======================================================\n\n\n\n\n"


======= iperf SERVER ON HOST_5 (on a different port==========



------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  6] local 10.0.0.5 port 5001 connected with 10.0.0.1 port 37488
[ ID] Interval       Transfer     Bandwidth
[  6]  0.0-10.0 sec  71.2 GBytes  61.0 Gbits/sec
[  6] local 10.0.0.5 port 5001 connected with 10.0.0.1 port 47722
[  6]  0.0-10.0 sec  71.6 GBytes  61.4 Gbits/sec
^C

======================================================




root@arun:/home/arun/mininet# printf "\n\n======= iperf SERVER ON HOST_5 (on a different port) ==========\n\n\
n\n" && iperf -s -p 7001 && printf "\n\n======================================================\n\n\n\n\n"


======= iperf SERVER ON HOST_5 (on a different port) ==========



------------------------------------------------------------
Server listening on TCP port 7001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  6] local 10.0.0.5 port 7001 connected with 10.0.0.1 port 36932
[ ID] Interval       Transfer     Bandwidth
[  6]  0.0-10.0 sec  70.3 GBytes  60.3 Gbits/sec
[  6] local 10.0.0.5 port 7001 connected with 10.0.0.1 port 40326
[  6]  0.0-10.0 sec  74.2 GBytes  63.6 Gbits/sec
^C

======================================================



root@arun:/home/arun/mininet# 
```

Fig. 6

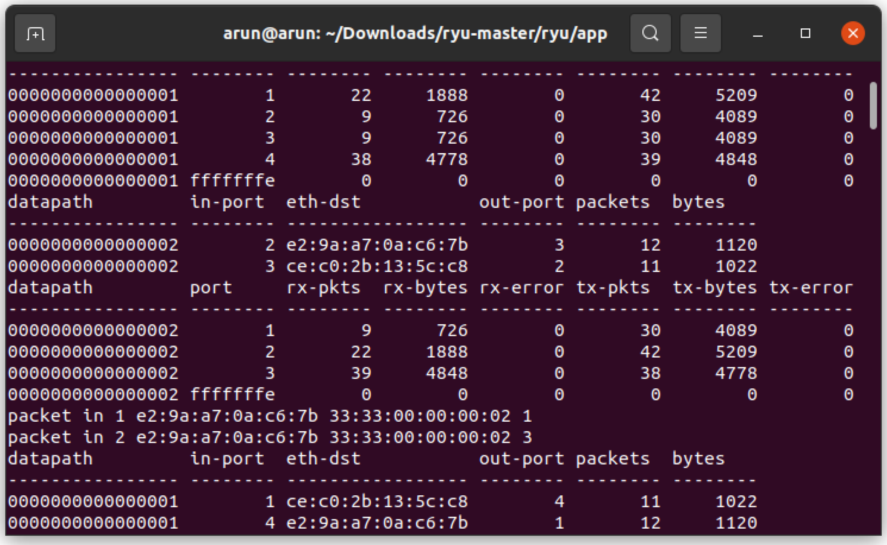
No packets were dropped for the *pingall* command in either of the implementations (Fig. 7)



```
Setting remote controller to 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s2) (h5, s2) (s1, s2)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
mininet> xterm h1 h2 h5
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5
h2 -> h1 h3 h4 h5
h3 -> h1 h2 h4 h5
h4 -> h1 h2 h3 h5
h5 -> h1 h2 h3 h4
*** Results: 0% dropped (20/20 received)
mininet> 
```

Fig. 7

Q2. The traffic monitor as a part of the learning switch was implemented (*firewall_monitor.py*). The traffic between h1 and h5 as a result of the ping requests can be seen from the statistics for the in-out ports in the console (Fig. 8).



```
---------------- -------- -------- -------- -------- -------- -------- --------
0000000000000001        1       22     1888        0       42     5209        0
0000000000000001        2        9      726        0       30     4089        0
0000000000000001        3        9      726        0       30     4089        0
0000000000000001        4       38     4778        0       39     4848        0
0000000000000001 fffffffe        0        0        0        0        0        0
datapath         in-port  eth-dst           out-port packets  bytes
---------------- -------- ----------------- -------- -------- --------
0000000000000002        2 e2:9a:a7:0a:c6:7b        3       12     1120
0000000000000002        3 ce:c0:2b:13:5c:c8        2       11     1022
datapath         port     rx-pkts  rx-bytes rx-error tx-pkts  tx-bytes tx-error
---------------- -------- -------- -------- -------- -------- -------- --------
0000000000000002        1        9      726        0       30     4089        0
0000000000000002        2       22     1888        0       42     5209        0
0000000000000002        3       39     4848        0       38     4778        0
0000000000000002 fffffffe        0        0        0        0        0        0
packet in 1 e2:9a:a7:0a:c6:7b 33:33:00:00:00:02 1
packet in 2 e2:9a:a7:0a:c6:7b 33:33:00:00:00:02 3
datapath         in-port  eth-dst           out-port packets  bytes
---------------- -------- ----------------- -------- -------- --------
0000000000000001        1 ce:c0:2b:13:5c:c8        4       11     1022
0000000000000001        4 e2:9a:a7:0a:c6:7b        1       12     1120
```

Fig. 8